

A MACAULAY2 PACKAGE FOR COMPUTATIONS WITH RATIONAL MAPS

GIOVANNI STAGLIANÒ

ABSTRACT. The Macaulay2 package *Cremona* performs some computations on rational and birational maps between irreducible projective varieties. For instance, it provides methods to compute degrees and projective degrees of rational maps without any theoretical limitation, from which is derived a general method to compute the push-forward to projective space of Segre classes. Moreover, the computations can be done both deterministically and probabilistically. We give here a brief description of the methods and algorithms implemented.

INTRODUCTION

In this note, we describe the computational package *Cremona* included with Macaulay2 since version 1.9 [GS17]. A first rudimentary version of this package has been already used in an essential way in [Sta15] (it was originally named *bir.m2*). Here we describe the version 3.0 of the package included with the current development version of Macaulay2.

Cremona performs computations on rational and birational maps between absolutely irreducible projective varieties over a field \mathbb{K} . Among other things, it provides general methods to compute projective degrees of rational maps, from which, as is well-known (see Proposition 1.2), one can interpret them as methods to compute the push-forward to projective space of Segre classes. The algorithms are naively derived from the mathematical definitions, with the advantages of being obvious, quite general and easily implemented. Moreover, all the methods (where this may make sense) are available both in a probabilistic version and in a deterministic version, and one can switch from one to the other with a boolean option named *MathMode*.

In Section 1, we will describe the main methods provided by the package and the algorithms implemented. Most of these have already been described in [Sta15, Section 2], but here we will consider a more general setting. For instance, Algorithm 1.3 for computing homogeneous components of kernels of homogeneous ring maps was presented in [Sta15, Algorithm 2.5] requiring that the map was between polynomial rings. In section 2, we will show how these methods work in some particular examples, concluding with an experimental comparison of the running times of one of these methods with the corresponding ones proposed in [Hel16] and [Har17] (see also [Jos15]).

For further technical details we refer to the documentation of the package, which can be shown using the command `viewHelp Cremona`.

1. DESCRIPTION OF THE MAIN METHODS

Throughout, we shall use the following notation. \mathbb{K} will denote a field; in practical, it can be for instance \mathbb{Q} , a finite field, or a fraction field of a polynomial ring over these. Let $\phi :$

Date: January 20, 2017.

2010 Mathematics Subject Classification. 14E05, 14Q15 .

Key words and phrases. Rational map, birational map, projective degrees, Segre class.

$X \dashrightarrow Y$ be a rational map from a subvariety $X = V(I) \subseteq \mathbb{P}^n = \text{Proj}(\mathbb{K}[x_0, \dots, x_n])$ to a subvariety $Y = V(J) \subseteq \mathbb{P}^m = \text{Proj}(\mathbb{K}[y_0, \dots, y_m])$, which can be represented, although not uniquely, by a homogeneous ring map $\phi : \mathbb{K}[y_0, \dots, y_m]/J \rightarrow \mathbb{K}[x_0, \dots, x_n]/I$ of quotients of polynomial rings by homogeneous ideals. Sometimes we will denote by $F_0, \dots, F_m \in \mathbb{K}[x_0, \dots, x_n]$ homogeneous forms of the same degree such that $\bar{F}_i := F_i + I = \phi(y_i)$, for $i = 0, \dots, m$. The common degree of these elements will be denoted by δ .

1.1. From algebraic geometry to computational algebra. For each homogeneous ideals $\mathfrak{a} \subseteq \mathbb{K}[x_0, \dots, x_n]/I$ and $\mathfrak{b} \subseteq \mathbb{K}[y_0, \dots, y_m]/J$, we have closed subschemes $V(\mathfrak{a}) \subseteq X$ and $V(\mathfrak{b}) \subseteq Y$, and the following basic formulae hold:

$$(1.1) \quad \overline{\phi(V(\mathfrak{a}))} = V(\phi^{-1}(\mathfrak{a})) \quad \text{and} \quad \overline{\phi^{-1}(V(\mathfrak{b}))} = V((\phi(\mathfrak{b})) : (\phi(y_0), \dots, \phi(y_m))^\infty).$$

In particular, we have that the (closure of the) image of ϕ is defined by the kernel of ϕ . Several issues about rational maps require to look for the left-hand sides of (1.1), and both the right-hand sides of (1.1) can be determined using Gröbner basis techniques, whenever \mathfrak{a} and \mathfrak{b} are explicitly given. Furthermore, Macaulay2 provides useful commands as `preimage`, `kernel` and `saturate`, so that the required users' skills are quite low. The aim of the package *Cremona* is to provide further tools.

1.2. Computing projective degrees. The projective degrees are the most basic invariants of a rational map. Many others can be derived from them, as for instance the dimension and the degree of the base locus. For more details on the subject, see [Har92, Example 19.4, p. 240].

Definition 1.1 (*Projective degrees*, [Har92]).

- (1) The projective degrees $d_0(\phi), d_1(\phi), \dots, d_{\dim X}(\phi)$ of the map ϕ are defined as the multidegree of the closure of the graph $\Gamma_\phi \subset \mathbb{P}^n \times \mathbb{P}^m$; equivalently,
- (2) the i -th projective degree $d_i(\phi)$ can be defined in terms of dimension and degree of the closure of $\phi^{-1}(L)$, where L is a general $(m - \dim X + i)$ -dimensional linear subspace of \mathbb{P}^m ; precisely, $d_i(\phi) = \deg \phi^{-1}(L)$ if $\dim \phi^{-1}(L) = i$, and $d_i(\phi) = 0$ otherwise.

In common computer algebra systems such as Macaulay2, it is easy to translate Definition 1.1 into codes. We now describe more in details how this can be done. All of this is implemented in the method `projectiveDegrees`; see Example 2.2 for an example using it.

1.2.1. Deterministic approach. Taking into account Definition 1.1(1), a bihomogeneous ideal for Γ_ϕ in $\mathbb{K}[x_0, \dots, x_n, y_0, \dots, y_m]$ can be for instance obtained as

$$(1.2) \quad (I + (\{y_i F_j - y_j F_i, 0 \leq i, j \leq m\})) : (F_0, \dots, F_m)^\infty.$$

Therefore its multidegree can be computed in Macaulay2 with the command `multidegree`, which implements an algorithm according to [MS05, p. 165].

1.2.2. Probabilistic approach. (See also [Sta15, Remark 2.4].) Taking into account Definition 1.1(2), if L is defined by an ideal I_L , the second formula of (1.1) says us that $\overline{\phi^{-1}(L)}$ is defined by the saturation of the ideal $(\phi(I_L))$ by $(\bar{F}_0, \dots, \bar{F}_m)$ in the ring $\mathbb{K}[x_0, \dots, x_n]/I$. So replacing the word *general* with *random* in the definition, we get a probabilistic algorithm that computes all the projective degrees. Moreover, we can considerably speed up this algorithm by taking into account two remarks: firstly, the saturation $\phi(I_L) : (\bar{F}_0, \dots, \bar{F}_m)^\infty$ is the same as $\phi(I_L) : (\lambda_0 \bar{F}_0 + \dots + \lambda_m \bar{F}_m)^\infty$, where $\lambda_0, \dots, \lambda_m \in \mathbb{K}$ are general scalars; secondly, the i -th projective

degree of ϕ coincides with the $(i - 1)$ -th projective degree of the restriction of ϕ to a general hyperplane section of X .

1.2.3. An alternative deterministic approach. Replacing the word *general* with *symbolic* in Definition 1.1(2) gives us a deterministic algorithm for computing projective degrees. For instance, in the case in which $\phi : \mathbb{P}^n \dashrightarrow \mathbb{P}^n$ is a dominant rational map, extending \mathbb{K} to the fractional field of a polynomial ring $\mathbb{K}[a_0, \dots, a_n]$, we have that $d_0(\phi)$ is the degree of the fibre of ϕ at the *symbolic point* $[a_0, \dots, a_n]$.

1.3. Some applications using projective degrees.

1.3.1. The degree of a rational map. The degree of the map $\phi : X \dashrightarrow Y$ is the number of isolated points in the inverse image of a general point of $\overline{\phi(X)}$ over the algebraic closure of \mathbb{K} . This is the same as the ratio of $d_0(\phi)$ and $\deg \overline{\phi(X)}$, and thus it can be explicitly computed. Let us note, however, that in several cases we do not need to compute the kernel of ϕ . For instance, if X is a projective space, we are able to pick up an abundance of rational points of $\overline{\phi(X)}$ and then we apply the second formula of (1.1). Another special case is when $d_0(\phi)$ is a prime number: here we have only to establish if the image of ϕ is a linear subspace (e.g. applying Algorithm 1.3 with $d = 1$). The method provided by *Cremona* for this computation is named `degreeOfRationalMap`. Related methods to this are `isBirational` and `isDominant` with obvious meaning. The latter does not compute the kernel of ϕ , but it uses an algorithm based on the fibre dimension theorem, and so using again the second formula of (1.1); this turns to be very powerful even in its deterministic version (see Example 2.1).

1.3.2. The Segre class. It is well-known that one can deduce an algorithm computing the push-forward to projective space of Segre classes from an algorithm computing projective degrees of rational maps between projective varieties and vice versa. Indeed, with our notation, we have the following:

Proposition 1.2 (Proposition 4.4 in [Ful84]; see also Subsection 2.3 in [Dol11] and Section 3 in [Alu03]). *Let $\mathfrak{B} \subset X$ be the subscheme defined by $\bar{F}_0, \dots, \bar{F}_m$ and let $v : X \hookrightarrow \mathbb{P}^n$ be the inclusion. If H denotes the hyperplane class of \mathbb{P}^n and $r = \dim X$, then the push-forward $v_*(s(\mathfrak{B}, X))$ of the Segre class of \mathfrak{B} in X is*

$$(1.3) \quad v_*(s(\mathfrak{B}, X)) = \sum_{k=0}^{\dim \mathfrak{B}} \left((-1)^{r-k-1} \sum_{i=0}^{r-k} (-1)^i \binom{r-k}{i} \delta^{r-k-i} d_{r-i}(\phi) \right) H^{n-k}.$$

The general method `SegreClass`, provided by *Cremona* for computing the push-forward to projective space of Segre classes, does basically nothing more than apply (1.3); see Example 2.3 for an example using this method. Furthermore, applying one of the main results in [Alu03], it is derived a method to compute the push-forward to projective space of the Chern-Schwartz-MacPherson class $c_{SM}(W)$ of the support of a projective scheme W ; recall that the component of dimension 0 of $c_{SM}(W)$ is the topological Euler characteristic of the support of W .

1.4. Computing homogeneous components of kernels. To compute, using *Macaulay2*, the homogeneous component of degree d of the kernel of ϕ , one can perform the command `ideal image basis(d, kernel phi)`. This is equivalent to the command `kernel(phi, d)` provided by *Cremona*, but the latter uses the following obvious algorithm.

Algorithm 1.3.

Input: the ring map ϕ and an integer d .

Output: homogeneous component of degree d of the kernel of ϕ .

- Find vector space bases G_0, \dots, G_r of $(\mathbb{K}[y_0, \dots, y_m]/J)_d$ and H_0, \dots, H_s of $I_{d\delta}$, where subscripts stand for homogeneous components;
- take generic linear combinations $\mathbf{G} = \sum_{i=0}^r a_i G_i$ and $\mathbf{H} = \sum_{j=0}^s b_j H_j$, and find a basis of solutions for the homogeneous linear system obtained by imposing that the polynomial $\mathbf{G}(F_0, \dots, F_m) - \mathbf{H} \in \mathbb{K}[a_0, \dots, a_r, b_0, \dots, b_s][x_0, \dots, x_n]$ vanishes identically;
- for each vector $(\hat{a}_0, \dots, \hat{a}_r, \hat{b}_0, \dots, \hat{b}_s) \in \mathbb{K}^{r+s+2}$ obtained in the previous step, replace in \mathbf{G} the coefficients a_0, \dots, a_r with $\hat{a}_0, \dots, \hat{a}_r$; return all these elements.

For small value of d , applying Algorithm 1.3 may turn out to be much faster than computing the whole kernel of the map; see for instance Example 2.1 below.

1.5. Inverting birational maps. General algorithms for inverting birational maps are known. One of them is implemented in the package *Parametrization* by J. Boehm, and the method `invertBirMap` of *Cremona* uses the same one for the general case as well. However, when the source X of the rational map ϕ is a projective space and a further technical condition is satisfied, then it uses the following powerful algorithm.

Algorithm 1.4 ([RS01]; see also [Sim04]).

Input: the ring map ϕ (assuming that ϕ is birational and further conditions are satisfied).

Output: a ring map representing the inverse map of ϕ .

- Find generators $\{(L_{0,j}, \dots, L_{m,j})\}_{j=1, \dots, q}$ for the module of linear syzygies of (F_0, \dots, F_m) ;
- compute the Jacobian matrix Θ of the bihomogeneous forms $\{\sum_{i=0}^m y_i L_{i,j}\}_{j=1, \dots, q}$ with respect to the variables x_0, \dots, x_n and consider the map of graded free modules $(\Theta \bmod J) : (k[y_0, \dots, y_m]/J)^{n+1} \rightarrow (k[y_0, \dots, y_m]/J)^q$;
- return the map defined by a generator $G = (G_0, \dots, G_n)$ for the kernel of $(\Theta \bmod J)$.

1.5.1. Heuristic approach. The method `approximateInverseMap` provides a heuristic approach to compute the inverse of a birational map modulo a change of coordinate. The idea of the algorithm is to try to construct the base locus of the inverse by looking for the images of general linear sections. Consider, for simplicity, the case in which $\phi : \mathbb{P}^n \dashrightarrow \mathbb{P}^{n'}$ is a Cremona transformation. Then, by taking the images of $n+1$ general hyperplanes in \mathbb{P}^n , we form a linear system of hypersurfaces in $\mathbb{P}^{n'}$ of degree $d_1(\phi)$ which defines a rational map $\psi : \mathbb{P}^{n'} \dashrightarrow \mathbb{P}^n$ such that $\psi \circ \phi$ is a (linear) isomorphism; i.e. we find an *approximation* of ϕ^{-1} . Next, we can fix the *error of the approximation* by observing that we have $\phi^{-1} = (\psi \circ \phi)^{-1} \circ \psi$. It is surprising that this method turns to be very effective in examples where other deterministic algorithms seem to run endlessly; see for instance Example 2.1 below.

2. EXAMPLES

In this section, we show how the methods described in Section 1 can be applied in some particular examples. We start with an example reviewing the construction given in [Sta15] of a quadro-quadric Cremona transformation of \mathbb{P}^{20} . Notice that we will omit irrelevant output lines.

Example 2.1. The code below constructs a ring map `psi` representing a rational map $\psi : \mathbb{P}^{16} \dashrightarrow \mathbb{P}^{20}$.¹ For this first part, we use the package *Cremona* only to shorten the code.

```
Macaulay2, version 1.9.2.1
with packages: ConwayPolynomials, Elimination, IntegralClosure, LLLBases,
               PrimaryDecomposition, ReesAlgebra, TangentCone
i1 : loadPackage "Cremona";
i2 : (Cremona.Options.Version, Cremona.Options.Date)
o2 = (3.0, Jan 11, 2017)
i3 : K = ZZ/70001;
i4 : P7 = K[t_0..t_7];
i5 : time E = saturate(minors(2, genericMatrix(P7, 4, 2)) + ideal sum(
    first entries gens image basis(2, ideal(t_0..t_3)), u -> random(K)*u), ideal(t_0..t_3));
    -- used 0.00413442 seconds
i6 : P8 = K[t_0..t_8];
i7 : time phi = toMap(sub(E, P8) + ideal(t_8), 2);
    -- used 0.00675865 seconds
i8 : time psi = toMap(kernel(phi), 2);
    -- used 0.142838 seconds
```

Up to this point, the computation was standard. But now we want to determine the homogeneous ideal of $Z := \overline{\psi(\mathbb{P}^{16})} \subset \mathbb{P}^{20}$, which turns out to be generated by quadrics. Computing this using `kernel psi` seems an impossible task, but it is elementary using `kernel(psi, 2)`. So we can consider ψ as a dominant rational map $\psi : \mathbb{P}^{16} \dashrightarrow Z \subset \mathbb{P}^{20}$.

```
i9 : time Z = kernel(psi, 2);
    -- used 2.57189 seconds
i10 : time psi = toMap(psi, Dominant=>Z);
    -- used 0.366501 seconds
```

The map ψ turns out to be not only dominant but birational.

```
i11 : time degreeOfRationalMap psi
    -- used 1.56033 seconds
o11 = 1
```

We now want to compute the inverse of ψ . This is a case where `invertBirMap` can apply Algorithm 1.4, but the running time is several hours. We can perform this computation fast enough calling appropriately the method `approximateInverseMap`.

```
i12 : time psi' = approximateInverseMap(psi, CodimBsInv=>10, MathMode=>true);
    -- used 15.5788 seconds
```

A Cremona transformation ω of \mathbb{P}^{20} is then obtained combining ψ^{-1} and Z as follows.

```
i13 : time omega = toMap(lift(matrix psi', ring Z) | gens Z);
    -- used 0.00438104 seconds
```

Even only to check the dominance of ω , performing for instance `kernel omega == 0`, it seems an impossible task, but this is elementary using `isDominant`.

```
i14 : time isDominant(omega, MathMode=>true)
    -- used 0.0332462 seconds
o14 = true
```

The dominance of ω can also be checked probabilistically with the following command.

¹Precisely, the algorithm for constructing ψ is as follows: take $E \subset \mathbb{P}^7$ to be a 3-dimensional Edge variety of degree 7, namely, the residual intersection of $\mathbb{P}^1 \times \mathbb{P}^3 \subset \mathbb{P}^7$ with a general quadric in \mathbb{P}^7 containing one of the \mathbb{P}^3 's of the rulings of $\mathbb{P}^1 \times \mathbb{P}^3 \subset \mathbb{P}^7$; next, see $E \subset \mathbb{P}^7$ embedded in a hyperplane of \mathbb{P}^8 and take the birational map $\phi : \mathbb{P}^8 \dashrightarrow \mathbb{P}^{16}$ defined by the quadrics of \mathbb{P}^8 containing E ; take $\psi : \mathbb{P}^{16} \dashrightarrow \mathbb{P}^{20}$ to be the map defined by the quadrics of \mathbb{P}^{16} containing the image of ϕ .

```
i15 : time degreeOfRationalMap omega > 0
      -- used 0.175495 seconds
o15 = true
```

We now check that our map is birational and compute its inverse using Algorithm 1.4.

```
i16 : time isBirational omega
      -- used 0.0361905 seconds
o16 = true
i17 : time omega' = invertBirMap omega;
      -- used 0.115647 seconds
i18 : time isInverseMap(omega, omega')
      -- used 0.366931 seconds
o18 = true
```

Finally, we determine some projective degrees (we are unable to calculate the complete list).

```
i19 : time projectiveDegrees(omega, 0)
      -- used 0.0336028 seconds
o19 = 1
i20 : time projectiveDegrees(omega, 1)
      -- used 0.074029 seconds
o20 = 2
i21 : time projectiveDegrees(omega, 18)
      -- used 3.97571 seconds
o21 = 4
i22 : time projectiveDegrees(omega, 19)
      -- used 0.0209049 seconds
o22 = 2
```

Example 2.2. In this example, we use the probabilistic versions of the methods. Take M to be a generic 3×5 matrix of linear forms on \mathbb{P}^6 , and let $\phi : \mathbb{P}^6 \dashrightarrow \mathbb{G}(2, 4) \subset \mathbb{P}^9$ be the rational map defined by the 3×3 minors of M (its base locus is a smooth threefold scroll over a plane).

```
i23 : P6 = K[x_0..x_6];
i24 : M = matrix pack(5, for i from 1 to 15 list random(1, P6));
i25 : -- the method toMap calls kernel(phi, 2)
      time phi = toMap(minors(3, M), Dominant=>2);
      -- used 2.39606 seconds
```

We check that the map is birational and compute its inverse.

```
i26 : time isBirational phi
      -- used 0.221249 seconds
o26 = true
i27 : time psi = invertBirMap phi;
      -- used 1.39805 seconds
```

Next we compute the multidegrees of ϕ and ϕ^{-1} . (By convention, the list of projective degrees is written in reverse order.)

```
i28 : time projectiveDegrees phi
      -- used 0.494998 seconds
o28 = {1, 3, 9, 17, 21, 15, 5}
i29 : time projectiveDegrees psi
      -- used 0.97424 seconds
o29 = {5, 15, 21, 17, 9, 3, 1}
```

Now we compute the push-forward to \mathbb{P}^6 (resp. \mathbb{P}^9) of the Segre class of the base locus of ϕ (resp. ϕ^{-1}) in \mathbb{P}^6 (resp. in $\mathbb{G}(2, 4)$).

```
i30 : time SegreClass phi -- element of ZZ[H]/H^7
-- used 0.427156 seconds
```

```
      6      5      4      3
o30 = - 680H + 228H - 60H + 10H
```

```
i31 : time SegreClass psi -- element of ZZ[H]/H^10
-- used 0.999509 seconds
```

```
      9      8      7      6      5
o31 = 728H - 588H + 276H - 98H + 24H
```

Finally, we check dominance and birationality of ψ .

```
i32 : time isDominant psi
-- used 0.101583 seconds
```

```
o32 = true
```

```
i33 : time degreeOfRationalMap psi
-- used 0.105057 seconds
```

```
o33 = 1
```

Example 2.3. In this example, we use the deterministic version of the method `SegreClass`. We take $Y \subset \mathbb{P}^{11}$ to be the dual quartic hypersurface of $\mathbb{P}^1 \times Q^4 \subset \mathbb{P}^{11*}$, where $Q^4 \subset \mathbb{P}^5$ is a smooth quadric hypersurface, and take $X \subset Y$ to be the singular locus of Y . We then compute the push-forward to the Chow ring of \mathbb{P}^{11} of the Segre class both of X in Y and of X in \mathbb{P}^{11} working over the Galois field $\text{GF}(331^2)$.

```
i34 : P11 = GF(331^2)[x_0..x_11];
```

```
i35 : Y = ideal sum(first entries gens minors(2, genericMatrix(P11, 6, 2)), t -> t^2);
```

```
i36 : X = sub(ideal jacobian Y, P11/Y);
```

```
i37 : time SegreClass(X, MathMode=>true) -- push-forward of s(X,Y)
-- used 4.08981 seconds
```

```
      11      10      9      8      7      6      5      4      3
o37 = 507384H - 137052H + 35532H - 9018H + 2340H - 658H + 204H - 64H + 16H
```

```
i38 : time SegreClass(lift(X, P11), MathMode=>true) -- push-forward of s(X, P^11)
-- used 5.76006 seconds
```

```
      11      10      9      8      7      6      5      4      3
o38 = 313568H - 101712H + 30636H - 8866H + 2532H - 720H + 198H - 48H + 8H
```

Example 2.4. Here we experimentally measure the probability of obtaining an incorrect answer using the probabilistic version of the method `projectiveDegrees` with a simple example of a birational map $\phi : \mathbb{G}(1, 3) \dashrightarrow \mathbb{P}^4$.

```
i39 : phi = (K) -> ( -- given K, returns phi defined over K
  R:=K[vars(0..4)];
  invertBirMap toMap(minors(2, matrix{(gens R)_0..3}, (gens R)_1..4}), Dominant=>2));
```

```
i40 : time m = projectiveDegrees(phi(QQ), MathMode=>true)
-- used 0.101609 seconds
```

```
o40 = {2, 4, 4, 2, 1}
```

```
i41 : pr = (K) -> ( -- given K, returns the probability that projectiveDegree phi(K) != m
  f:=phi(K); p:=0; for i to 999 do if projectiveDegrees f != m then p=p+1;
  toString(p*0.1) || "%");
```

```
i42 : time pr(QQ) -- probability when K = QQ
-- used 60.7742 seconds
```

```
o42 = 0%
```

```
i43 : time pr(ZZ/70001) -- probability when K = ZZ/70001
-- used 44.5745 seconds
```

```
o43 = 0%
```

```
i44 : time pr(GF(3^8)) -- probability when K = GF(3^8)
```



```

-- used 64.3963 seconds
o44 = .3%
i45 : time pr(ZZ/101) -- probability when K = ZZ/101
-- used 48.9237 seconds
o45 = 8.8%
i46 : time pr(ZZ/31) -- probability when K = ZZ/31
-- used 48.7806 seconds
o46 = 27.2%

```

Example 2.5. In this last example, we deal with an experimental comparison of the method `SegreClass` of *Cremona* and the corresponding ones of other Macaulay2 packages. To our knowledge, the only other deterministic method to be able to compute Segre classes is implemented in the package *CSM-A*, by P. Aluffi (see [Alu03]). This method requires the ambient variety to be a projective space, and in this case there should be no significant differences with respect to the deterministic version of `SegreClass`. We then want to compare the probabilistic version of `SegreClass` against the corresponding methods of the two following packages: *CharacteristicClasses*, by M. Helmer and C. Jost (see [Hel16, Jos15]); and *FMPIntersectionTheory*, by C. Harris (see [Har17]). Since the former puts restrictions on the ambient variety, we will consider only examples where the ambient is a projective space. We are unable to determine precisely which is the fastest among the three methods and which has highest probability of giving the correct answer. We just propose below some examples; the running times of these with the addition of a few others are summarized in Table 1.

```

i47 : loadPackage "CharacteristicClasses";
i48 : (CharacteristicClasses.Options.Version, CharacteristicClasses.Options.Date)
o48 = (2.0, October 24, 2015)
i49 : loadPackage "FMPIntersectionTheory";
i50 : (FMPIntersectionTheory.Options.Version, FMPIntersectionTheory.Options.Date)
o50 = (0.1, February 9, 2015)

```

Running times for computing the Segre class of the Veronese surface in \mathbb{P}^5 over $\mathbb{Z}/3331$.

```

i51 : X = trim minors(2, genericSymmetricMatrix(ZZ/3331[vars(0..5)], 3));
i52 : time SegreClass X; -- probabilistic method in Cremona
-- used 0.039288 seconds
i53 : time Segre X; -- method in CharacteristicClasses
-- used 0.402183 seconds
i54 : time segreClass X; -- method in FMPIntersectionTheory
-- used 0.0737198 seconds
i55 : time SegreClass(X, MathMode=>true); -- deterministic method in Cremona
-- used 0.0136062 seconds

```

Running times for computing the Segre class of a generic complete intersection of type $(2, 3, 3)$ in \mathbb{P}^5 over $\mathbb{Z}/16411$.

```

i56 : X = last(P5=ZZ/16411[vars(0..5)], ideal(random(2,P5), random(3,P5), random(3,P5)));
i57 : time SegreClass X; -- probabilistic method in Cremona
-- used 0.0972829 seconds
i58 : time Segre X; -- method in CharacteristicClasses
-- used 0.697108 seconds
i59 : time segreClass X; -- method in FMPIntersectionTheory
-- used 0.308239 seconds
i60 : time SegreClass(X, MathMode=>true); -- deterministic method in Cremona
-- used 9.9309 seconds

```

Running times for computing the Segre class of a rational normal scroll $S(1, 4) \subset \mathbb{P}^6$ over \mathbb{Q} .

```

i61 : X=last(P6=QQ[x_0..x_6], minors(2, matrix{{x_0, x_2..x_5}, {x_1, x_3..x_6}}));
i62 : time SegreClass X; -- probabilistic method in Cremona

```



```

-- used 0.694507 seconds
i63 : time Segre X;      -- method in CharacteristicClasses
-- used 1.05042 seconds
i64 : time segreClass X; -- method in FMPIntersectionTheory
-- used 4.05631 seconds
i65 : time SegreClass(X,MathMode=>true); -- deterministic method in Cremona
-- used 0.0819738 seconds

```

Input	Characteristic- Classes.m2	FMPIntersection- Theory.m2	Cremona.m2 (probabilistic)	Cremona.m2 (deterministic)
Veronese surface	0.40	0.07	0.04	0.01
complete int. of type $(2,3,3)$ in \mathbb{P}^5	0.70	0.31	0.10	9.93
rational normal surface in \mathbb{P}^6	1.05	4.06	0.69	0.08
Edge variety $E \subset \mathbb{P}^7$ in Ex. 2.1	0.06	0.18	0.07	0.20
base locus of ϕ in Ex. 2.2	0.21	6.49	0.43	1066.24
$X \subset \mathbb{P}^{11}$ in Ex. 2.3 over \mathbb{F}_{331^2}	49.81	—	85.12	5.76
$X \subset \mathbb{P}^{11}$ in Ex. 2.3 over $\mathbb{Z}/16411$	2.73	187.63	13.32	2.29

TABLE 1. Run-times to compute Segre classes (all times given in seconds)

REFERENCES

- [Alu03] P. Aluffi, *Computing characteristic classes of projective schemes*, J. Symbolic Comput. **35** (2003), no. 1, 3–19.
- [Dol11] I. Dolgachev, *Lectures on Cremona transformations*, Ann Arbor-Rome, available at <http://www.math.lsa.umich.edu/~idolga/cremonalect.pdf>, 2010/2011.
- [Ful84] W. Fulton, *Intersection theory*, Ergeb. Math. Grenzgeb. (3), no. 2, Springer-Verlag, 1984.
- [GS17] D. R. Grayson and M. E. Stillman, MACAULAY2 — *A software system for research in algebraic geometry (version 1.9.2)*, available at <http://www.math.uiuc.edu/Macaulay2/>, 2016.
- [Har92] J. Harris, *Algebraic geometry: A first course*, Grad. Texts in Math., vol. 133, Springer-Verlag, New York, 1992.
- [Har17] C. Harris, *Computing Segre classes in arbitrary projective varieties*, in press, doi:10.1016/j.jsc.2016.09.003, 2017.
- [Hel16] M. Helmer, *Algorithms to compute the topological Euler characteristic, Chern–Schwartz–Macpherson class and Segre class of projective varieties*, J. Symbolic Comput. **73** (2016), 120–138.
- [Jos15] C. Jost, *Computing characteristic classes and the topological Euler characteristic of complex projective schemes*, J. Softw. Algebra Geom. **7** (2015), no. 1, 31–39.
- [MS05] E. Miller and B. Sturmfels, *Combinatorial commutative algebra*, Grad. Texts in Math., vol. 227, Springer-Verlag, New York, 2005.
- [RS01] F. Russo and A. Simis, *On birational maps and Jacobian matrices*, Compos. Math. **126** (2001), no. 3, 335–358.
- [Sim04] A. Simis, *Cremona transformations and some related algebras*, J. Algebra **280** (2004), no. 1, 162–179.
- [Sta15] G. Staglianò, *Examples of special quadratic birational transformations into complete intersections of quadrics*, J. Symbolic Comput. **74** (2015), 635–649.

E-mail address: giovannistagliano@gmail.com